COMMICAST
Vol. 5, No. 1, March 2024, pp. 43-62
DOI: https://doi.org/10.12928/commicast.v5i1.9955
ISSN: 2723-7672

Bus*: An efficient algorithm for finding Moving K-Nearest Neighbors (MKNNs) with capacity constraints

[1]Saad Aljubayrin*

[1] Department of Computer Science, College of Computing and Information Technology, Shaqra University, Shaqra 11961, Saudi Arabia
[1] aljubayrin@su.edu.sa*
*Correspondent email author: aljubayrin@su.edu.sa

ARTICLE INFO

Article history
Received 2024-01-23
Revised 2024-03-15
Accepted 2024-03-25

Keywords
A*
Alogarithm
Bus Path
Moving k-Nearest Neighbor
Optimization

ABSTRACT

The large-scale and increasing use of transportation systems in various applications is expected to become an important component of communications networks beyond 5G and 6G in the next decade. To effectively support the massive deployment of transportation systems, reliable, secure, and cost-effective wireless connectivity is required. Communication networks are very important for vehicles that act as mobile user equipment. Although communications networks offer a promising way for cars to stay connected, it isn't easy to make transportation work well. This paper aims to present a new and interesting problem: the finding of the Moving K-nearest neighbors (MKNNs), where each neighbor has a capacity limit. Specifically, considering a set of moving objects with different capacity constraints distributed in the road network, query objects with a certain load, find the optimal set of neighbors where the total available capacity is equal to or greater than the load of the query object, and the total travel time of the optimal set to reach the query object is minimized. This problem has significant applications in our lives. For example, it can help bus operating companies find other optimal bus trains in operation to move to the location of the damaged bus and transport its passengers to their destinations. In contrast, the total travel time of the optimal train is minimized. This paper uses previous research methods with a qualitative descriptive approach from sources that researchers found. The results of this research serve as material for proposing new algorithms that are effective for solving problems in real-time when using real data sets.

1. Introduction

The next generation of communication networks, dubbed 6G, are expected to provide intelligent, secure, dependable, and limitless connectivity (Khan, Jamshed, et al., 2023),(Raza et al., 2022). 6G is expected to bring a full-fledged framework with integrated terrestrial and non- terrestrial networks for connected things and automation sys- tems, ranging from autonomous cars to unmanned aerial vehicles, with stringent and diverse requirements for reliabil- ity, latency, data rate, and energy efficiency (Khowaja et al., 2023),(Khan, Ali, et al., 2023). Next generation

transportation systems play critical roles in various use cases and scenarios extending beyond 5G and 6G (Ahmed, Raza, et al., 2022),(Mahmood, Vu, et al., 2022). The deployment of self-driving cars will skyrocket in the coming decades (Khan, Ihsan, et al., 2022). Other 6G technologies which can be integrated into next-generation transportation systems include intelligent reflecting surfaces (Ihsan et al., 2022), backscatter communications (Khan, Lagunas, et al., 2022), cognitive radio (Khan, Abbas, et al., 2022), non-orthogonal multiple access (Khan et al., 2021), artificial intelligence/machine learning (Jameel et al., 2019), the Inter- net of things (Khan et al., 2020), and millimeter wave/terahertz frequencies (Rasheed et al., 2023). Recently, researchers in industry and academia have been actively investigating different problems related to next- generation transportation systems (Asif et al., 2023),(Ihsan et al., 2023).

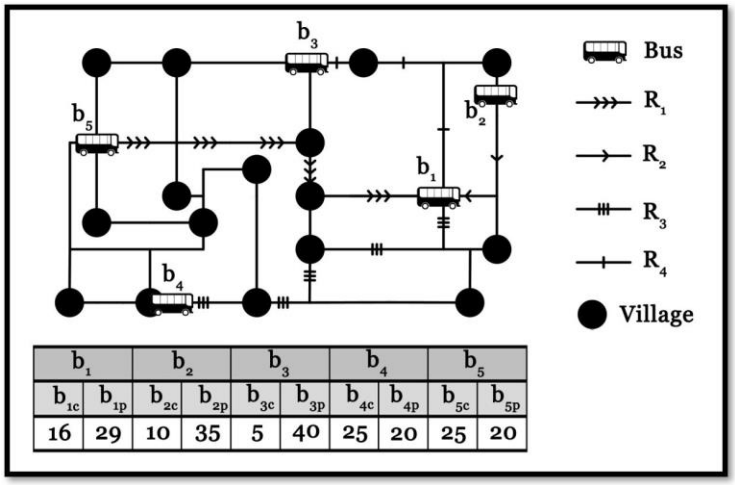
Vehicle to everything communications has piqued the in- terest of both academia and industry in recent years (Ali, Khan, et al., 2021). Vehicle to everything encompasses a wide range of wireless technologies as a key enabler for intelligent transportation systems, including vehicle to vehicle communications, vehicle to infrastructure communications, and vehicle to pedestrian communications, as well as communications with vulnerable road users and cloud networks (Khan, Jamshed, et al., 2022),(Ahmed, Khan, et al., 2022). The grand vision is that Vehicle to everything communications, enabled by 6G wireless systems will be an essential component of future connected autonomous vehicles (Khan et al., 2019),(Khan et al., 2021). Furthermore, Vehi- cle to everything communications will provide numerous far- reaching and game-changing benefits, including a completely new user experience, significant improvements in road safety and air quality, a diverse range of transportation applications and use cases, and numerous advanced applications (Ali, Farooq, et al., 2021),(Khan, Lagunas, Ali, et al., 2022). Next generation communications involve mobile edge computing (Mahmood et al., 2021), simultaneous wireless information and power transfer (Mahmood, Ahmed, et al., 2022), relay networks (Khan, 2019), heterogeneous networks (Khan, Li, et al., 2021), security and reliability (Hasan et al., 2023), device to device communications (Yu et al., 2021), green communication network (Mahmood et al., 2020), low powered sensors devices (Khan, Imtiaz, et al., 2021), cooperative communications (Ali et al., 2022), and satellite communications (Khan, Lagunas, Mahmood, Elhalawany, et al., 2022).

Finding K Nearest Neighbors problems have been in- vestigated extensively in the spatial and temporal database community for the past couple of decades, In both Euclidian (Hautamäki et al., 2004),(Athitsos et al., 2005) and spatial network (Shahabi et al., 2002),(Jensen et al., 2003) variants. This results in important outcomes in fields such as data clas- sification (Matke et al., 2023), POIs quires (Aljubayrin, He, et al., 2015) and urban planning (Jensen et al., 2003). In this paper, we focus on the spatial network variant and introduce a novel and interesting problem: finding Moving K-Nearest Neighbors with capacity constraints MKNNsCC query. In particular, given a road network

 https://doi.org/10.12928/commicast.v5i1.9955

N, a set of n moving buses B = {b1, b2, b3, · · · bn} with the available passengers capacity bic for each bus, a broken-down bus bx with a number of passengers bxp who need to reach their destinations; find the optimal set of buses OpB = {OpB1, OpB2, OpB3 · · · OpBn}inB to travel to bx and trans- port its passengers to their destinations, where the total capac- ity of OpB, OpBc = {OpB1c, OpB2c+OpB3c, · · · OpBnc} ≥ bxp and the total travel time of the optimal set OpBt to reach bx is minimized, OpBt = {OpB1t, OpB2t, · · · OpBnt} ≤ OpBmt = {OpB1t, OpB2t, · · · OpBmt} where OpBmt is any other possible set in B. To better illustrate the MKNNsCC query, we would like to first distinguish it from the traditional Moving k Nearest Neighbor (MkNN) query as it is presented in Nutanong et al., (2009).

Specifically, the MkNN query is defined as a continuously moving object s in a road network N, and a set of neighbor objects Nob = {Nob1, Nob2, Nob3, · · · Nobn} while the query objective is always to maintain the set of k objects, which are the closest to the query object x. For example, when an ambulance driver always wants to maintain the five nearest available emergency departments to deliver a patient. Another example is when a delivery service driver always wants to keep track of a list of the three nearest petrol stations while moving around the suburbs (Khan, Jameel, et al., 2020). When looking for the k nearest neighbors, the MKNNsCC query takes into account the capacity constraint as an addi- tional optimality dimension. This is the primary distinction between the MkNN query and the query that is presented in this paper. To be more specific, whereas the MKNN query will only find the k neighbors that are physically closest, our query will find the optimal set of neighbors by taking into account the capacity that is currently available for each neighbor (Petrescu-Mag et al., 2020). As a consequence of this, there is a chance that some of the neighbors who are closest to you will be eliminated due to their limited capacity. In the following example, we will explain how to use the MKNNsCC query (Jan et al., 2017).



b_1		b_2		b_3		b_4		b_5	
b_{1c}	b_{1p}	b_{2c}	b_{2p}	b_{3c}	b_{3p}	b_{4c}	b_{4p}	b_{5c}	b_{5p}
16	29	10	35	5	40	25	20	25	20

Fig. 1. Motivating Example

A sample of a road network is presented in Figure 1, along with several school buses that transport students to their respective villages. Each of the buses bi has two variables: the number

of passengers currently on the bus bip and the available passenger capacity bic. When one of the buses breaks down, there is an immediate need to transport its passengers to their destinations using one or more of the other operating buses. Determining the best group of needed buses depends on two main factors: the group's total available passenger capacity and the group's total traveling time. For example, when the bus b1, which is transporting 29 students b1p=29, breaks down, we need to find one or more buses to transport the 29 passengers to their destinations.

The bus b2 is very close to b1, yet its available capacity b2c=10, which is insufficient to transport the passengers of the broken bus b1p > b2c. Thus, we need to search for other buses in addition to b2 to transport the broken bus passengers b1p. Although, the bus b3 is relatively close to b1, it has a limited available passengers' capacity b3c=5. On the other hand, the bus b4, which is quite further to b1, has the advantage of a large capacity b4c=25. Therefore, the optimal set of buses OpB to transport the passenger of b1 are b2 and b4, OpB={b2,b4}.

It can be clearly seen from the example that determining the optimal set OpB does not only depend on the traveling time for each candidate to reach b1, but it also considers the capacity. For instance, the buses b4 and b5 have the same available capacity b4c = b4c=25 yet, the bus b4 is in the optimal set b4 ∈ OpB, while b5 / ∈ OpB This is because the route R3 from b4 to b4 is shorter than the route R1 from b5 to b5. Another possible example where the MKNNsCC query can be helpful is when a delivery truck breaks down while delivering goods, the truck operating company can use the MKNNsCC query to find the optimal set of other delivery trucks within close range with a sufficient capacity to deliver the goods of the broken-down truck. We formulated the MKNNsCC query after being inspired by situations that were comparable to the examples that came before it.

To answer the query, we proposed an algorithm that is both effective and efficient, and we called it Bus*. Finally, we used a real dataset to evaluate the algorithm's effectiveness and performance. In this paper, we used an offline framework to pre-compute the travel time between any two points in the bus network. This technique is commonly used in the special and temporal database community, as discussed in (Huang et al., 2007). We indexed the road network into a spatial data structure, pre-computed the average traveling time between every two nodes, and stored the real traveling time for different time slots during the day. The name of the proposed Bus* algorithm is inspired by the well-known A* algorithm (Hart et al., 1968).

In general, the Bus* algorithm is based on creating a virtual fully connected weighted graph G, where the nodes are the locations of the buses at the query time. In G, we assume there is an edge between every two nodes, and the weight of all edges connected to a node is the traveling time from that node to the query node (e.g., the broken Bus). The Bus* algorithm starts as a basic best-first

 https://doi.org/10.12928/commicast.v5i1.9955

search in G, where the search starts from the query node in all directions. The algorithm maintains a priority queue for the candidate set of discovered buses.

Whenever a new node is visited, we create new candidate sets based on the nodes' edges and add them to the queue along with two significant variables, the total traveling time and the total passengers' capacity of each candidate set (Sparrow, 2004). The candidate set with the lowest traveling time always starts the best-first search. The search stops once the needed capacity is reached and there are no other candidate sets with similar or better traveling times. The final returned candidate/s (e.g., if multiple sets of buses share the same traveling time) is the optimal set of buses OpB, where the order of the buses in the set is ineffective.

In this work, we present the following contribution: We introduce the MKNNsCC query, which has significant applications in our life. We proposed the novel Bus* algorithm to solve the MKNNsCC query and produce optimal results. Wecarried out extensive experiments to evaluate the effi ciency and accuracy of the Bus* algorithm, which shows high effective results and high performance compared to the baseline algorithm. The remainder of this paper is presented along these lines. Section 2 discusses the related work in Moving K-Nearest Neighbors (MKNNs) finding problems. Section 3 presents the preliminaries and defines the MKNNsCC query problem. In Section 4, we address the used solution framework and detail the Bus* algorithm and The experimental results are illustrated. Finally, in Section 5, we conclude the paper.

2. Theoretical Framework

In this section we interduce the Bus* algorithm, which solve the MKNNsCC query efficiently. The name of this algorithm is inspired by the well-known A* algorithm (Hart et al., 1968), which can be described as an enhanced version of Dijkstra's algorithm since it utilizes heuristics to lead the graph search. A* algorithm finds the shortest path between two nodes by exploring the most promising nodes starting from the source node. It also uses a priority queue to maintain all discovered nodes along with their shortest achieved paths. The search terminates ones the destination node is reached and there are no more promising nodes to explore. Similarly, the Bus* algorithm consists of two main stages: Bus* virtual graph creation stage and running the Bus* algorithm stage. In the f irst stage, we create a virtual fully connected graph where the nodes are the moving objects (buses) at the query time. In the second stage we run the Bus* algorithm, which uses a best-first searching technique, over the virtual graph until it terminates and returns the optimal set of buses.

Bus* Virtual Graph Creation Stage

In this stage we create a virtual fully connected graph G, where the nodes are the locations of all buses with at least one passenger possible capacity (bic > 1) in the road network at the query time. In G, we assume there is an edge between every two nodes and the weights of all edges connected

to a node are the travelling time from that node to the query node (e.g., the broken bus). The number of edges in G heavily depends on the number of nodes, which are the buses in the road network. The connectivity of G can be measured by n(n-1), where n is the number of nodes. The travelling time between any bus b' in the road network and the broken bus b is not computued at the query time, instead, we use the precomputed estimated time between the quadtree leaf node containing b' and the other leaf node containing b as discussed in the network travel time estimation framework.

For example, in Fig. 3, assume that b1 is the broken bus. First, we connect b1 with every other bus {b2,b3,b4,b5}. Next, we connect all the other buses {b2,b3,b4,b5} with each other by using directed edges, hence, there are two edges between any pair of buses. In this graph, the weights of all edges connected to a bus are similar and they are equal to the travelling time between that bus and the broken bus. For example, the weight of all edges connected to b2 is 4, although the edge connecting b5 with b2 seems longer than that connecting b3 with b2. In addition, the weight of the two edges between a pair of buses is different, thus, they cannot be replaced with a single edge. For instance, the weight of the edge from b2 to b3 is 4 while the weight of the edge from b3 to b2 is 6.

Bus* Running Stage

In this stage we introduce the term candidate set of buses CdB, which consists of a chain of buses and when completed can be returned as an answer to the MKNNsCC query. Each CdB is assigned with two significant variables: (1) the candidate set total travelling time CdBt reaching the broken bus and (2) the total passenger capacity of that candidate set CdBc. In addition, we need to construct a priority queue, where we add and prioritize the discovered candidate set of buses CdB. The priority of a candidate set is determined by its total travelling time CdBt. The Bus* algorithm starts as a basic best-first search in G, where the search starts from the query node towards all directions. For each edge of a new visited bus, we create a new candidate set CdB along with its variables (CdBc, CdBt) and add it to the priority queue as long as the set has not been added previously.

The candidate set with the lowest traveling time, CdBt always starts the best-first search and new candidates are added to the priority queue. The last bus added to CdB is the one leads the CdB expansion. This process keeps iterating until the total passenger capacity of a candidate set CdBc reaches the needed capacity of the broken bus and there are no other candidate sets with similar or better CdBt. The final returned candidate/s (e.g., if there are multiple sets of buses sharing the same traveling time) is the optimal set of buses OpB. The order of buses in the OpB is insignificant as they would move to the broken bus location simultaneously.

https://doi.org/10.12928/commicast.v5i1.9955

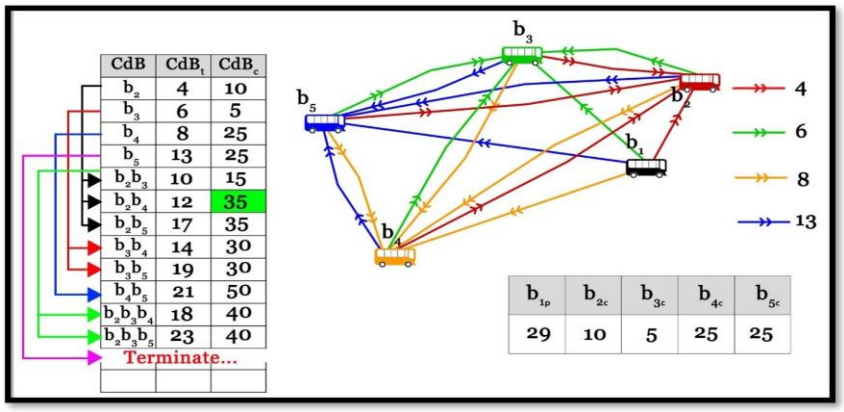


Fig. 2. Bus* Running Stage

For example, in Fig. 3 each of the 4 buses connected to b1 creates a new candidate set such that CdB1 = b2, 7 CdB2 = b3, CdB3 = b4, CdB4 = b5. Additionally, the CdBt and CdBc for each CdB can be extracted such that CdB1t=4 and CdB1c=10 as shown in the figure table. Next, we add the candidate sets to the priority queue, which sorts them based on their CdBt. Since none of the candidate sets total capacity CdBc meets the broken bus capacity b1p=29, we need a new iteration. In the new iteration, the candidate b2 will be processed as it has the lowest CdBt=4. As discussed above, the last added node to the candidate set is the one leading the set expansion, which applies to the node b2 in CdB{b2},therefore, it leads the search and creates three new candidate sets CdB{b2,b3}, CdB{b2,b4} and CdB{b2,b5}.

This process keeps iterating till it meet two conditions: (1) there is a candidate set with CdBc that satisfies the query con strain such that CdBc ≥ b1p (2) there is no more unexpanded candidate set with CdBt that is similar or less than the best found candidate set so far. These two conditions apply on the candidate set CdB={b2,b3} since CdBc(35) ≥ b1p (29) and there is no unexpanded candidate sets with CdBt ≥ {b2,b3}t (12).

Therefore, the search terminates before it expands the candidate set CdB={b5} and any further candidate sets. As shown in the previous example, the main benefit of the Bus* algorithm is its ability to detect the lack of any further promising candidate set, thus, terminates the search before exploring the fully connected graph G. This results in more efficient process and effective results as will be demonstrated in Section V.

3. Method

Research finding K Nearest Neighbors' problems have been an interesting area over the past few decades (Zhao et al., 2024). To the best of our knowledge, there is no previous attempt to investigate the problem of finding moving K-Nearest Neighbors problem with capacity constraints MKNNsCC. The previous related research can be categorized into two main categories: finding K nearest

neighbors in Euclidian space and finding K nearest neighbors in spatial networks. First, the majority of the existing studies have focused on f inding K nearest neighbors in Euclidian space.

For example, the study in Lopac et al., (1986) focuses on the different dimensions of the object when finding the nearest neighbor. Other examples are the papers presented in (Li et al., 2014)-(Basu et al., 2015), which all use the data structure Quad-tree to best find the nearest neighbor. In addition, the work in Duch & Martinez, (2005), which studies the range nearest neighbor query, is another instance of the Euclidian space approach.

In particular, the authors define a set of points (range) in d dimensional space as an input, while the output is all nearest neighbors to the input range. A possible application of their query is to find all the nearest hotels to a particular park. All the previous studies are different from the MKNNsCC query. This is because they do not consider the neighbors' capacity. Therefore, their solutions do not apply to the MKNNsCC query.

Next, we compare the second category of the related work, which finds K Nearest Neighbors in spatial networks, with the MKNNsCC query. The work in Shahabi et al., (2002) proves that the Euclidian distance metric cannot be directly applied to find KNNs in spatial networks as it returns inaccurate results. Instead, they convert a road network into high dimensional space and apply the Euclidian metrics to find the KNNs objects. The problem investigated in Shen et al., (2017) is similar to the previous work; nevertheless, they introduce a new index called V-tree to search the road network for KNNs efficiently.

The problem definition in the previous two works differs from our problem definition as we consider the capacity constraint. Another research related to this paper is Zheng & Su, (2014), where the authors utilize a non-parametric algorithm to forecast the value of a road network. In specific, they use the KNNs state vectors of a query state vector to forecast its traffic status in the short-term future. Their approach requires extensive and representative data for an accurate result. Again, the previous problem differs from ours as we are not interested in forecasting the road network; instead, we focus on finding the KNNs with capacity constraints.

In addition, the work in Tianyang et al., (2019) studies finding the KNNsobject in a road network while considering the direction of the NNs as a data quality constraint. They proposed an algorithm based on an R-tree index to eliminate the non promising NNs based on their direction. Although their work is based on a constraint, it is not the capacity constraint as in our query MKNNsCC. Thus, their solution is inapplicable to our problem. The most related research to our query is the work in Wang et al., (2018) where the author's study location the ideal dynamic interaction locations for multiple moving objects optimization problems. For example, when a group of friends from

 https://doi.org/10.12928/commicast.v5i1.9955

different work locations wants to find the optimal point to meet for a ride-sharing to a party (Assegaff & Pranoto, 2020).

Another example is when a group of friends wants to find the optimal POI (e.g., caf´ e) to meet while each of them is on her way home. The optimality of the chosen point is regarding the travel cost of all moving objects towards the meetup point while considering the road network constraints such as traffic conditions, road closure, weather, or the constraints of the moving object such as the continuous trip of each object is shown in the second example.

They proposed five methods and a constraint-based geoprocessing framework to tackle this problem. Although the previous study considers some constraints on a road network, it is different from than MKNNsCCproblem; thus, their solution is inapplicable to our query. This is because our query results in an optimal set of moving objects while they aim to find an optimal location for a set of moving objects. Moreover, our query takes the constraint of the moving object into account while the constraints in their problem are within the road network.

Table 1. FREQUENTLY USED NOTATIONS

Symbol	Explanation
MKNNsCC	Moving K-Nearest Neighbors with capacity constraints
N	Road network
B	Set of moving buses
B_c	Bus available capacity
B_{full}	Bus full capacity
B_t	Travel time to reach the broken-down bus
B_p	Number of a bus passengers
CdB	Candidate set of busses
CdB_c	Total capacity of a candidate set of buses
CdB_t	Total travel time of an optimal set of buses to reach the broken-down bus
OpB OpB_c OpB_t	Optimal set of buses Total capacity of an optimal set of buses Total travel time of an optimal set of buses to reach the broken-down bus

Following the presentation of the formalization of the MKNNsCC query and the introduction of the baseline al gorithm, we will then proceed to the presentation of the framework that is used to estimate the travel time. The most common notations are outlined in the table that is referenced as Table 1.

Problem Definition

Given a weighted road network N, a set of n moving school buses B = {b1,b2,b3,⋯bn} where each bus has a different passengers capacity. We denote a bus current available capacity with bic and it is based on the bus full capacity biFull and the number of onboard passengers bip, such that bic = biFull − bip. Let bx be a broken-down school bus with a number of passengers denoted as bxp who need to reach their destination. For example, in Figure 3 assume the broken-down bus is b1

and the driver of b1 called the nearest bus b2 to immediately travel to the breakdown location and transport b1 stuck passengers b1p to their villages. As shown in the figure, the number of the broken-down bus passengers b1p=29 and the capacity of the rescue bus b2c=10, thus, the immediate nearest neighbor bus is not always the best solution.

In addition, a single rescue bus might not be sufficient to solve the problem. To find a valid candidate bus b or a candidate set of busses B to rescue b1 we need a total capacity of at least 29 passengers. Moreover, Adding the second nearest neighbor to b1 which is b3 to b2 does not form a candidate set of busses and does not answer the query as b2c + b3c < b1p. On the other hand, adding b5 to b2 creates a valid candidate set of busses as b2c+ b5c > b1p. However, the set B = {b2,b5} is not necessarily the optimal set as there can be another set e.g., B = {b2,b4} which also satisfies the constrain B2c ≥ bip and its total travel time to reach b1; B2t −b2t+b4t is shorter and the total travel time of B1 such that B2t < B1t. Therefore, the optimal set of buses OpB to transport the passenger of b1 are b2 and b4 and OpB ={b2,b4}.

Definition 1 Moving K-Nearest Neighbors with Ca pacity Constraints MKNNsCC Query:: Given a road network N, a broken-down bus bx with a number of passengers bxp, a set of n moving buses B = {b1,b2,⋯bn} with differ ent capacities bic, the MKNNsCC query finds the optimal set of buses OpB = {OpB1,OpB2⋯OpBn} ∈ B to travel to bx, where the total capacity of OpB, OpBc = {OpB1c + OpBc2 + ⋯OpBcn} ≥ bxp and the total travel time of the optimal set OpBt to reach bx is mini mized OpBt = {OpB1t,OpB2t,⋯OpBnt} ≤ OpBmt = {OpB1t,OpB2t,...OpBmt} where OpBm is any other pos sible set in B , i.e., ∀OpBm,OpBnc ≤ OpBmVOpBnt < OpBm. Based on the above problem definition, a na¨ ıve solu tion would be first to find all possible busses combinations B in B excluding bx and compute the total capacity Bc and total travel time Bt for each combination B. Next, we eliminate the combinations with a total capacity that is less than the broken bus number of passengers e.g Bc < bxp. Finally, we sort the remaining combinations ascendingly based on the travel time of each combination Bt to find the combination per second with the lowest travel time to be the optimal set per sec OpB. The issue with the above discussed na¨ ıve solution is that it is inefficient in terms of the time consumed to process the MKNNsCC query. In addition, the complexity of this solution is O(2n) where n is the number of busses.

Network Travel Cost Estimation Framework

Using the Euclidian distance to estimate the travelling time in road network and find the nearest moving objects is an imprecise measurement (Shahabi et al., 2002),(Jensen et al., 2003)-(Aljubayrin, Qi, et al., 2015). In addition, using the abstract weight of the road network segments (e.g., distance) might not be always accurate (Aljubayrin, He, et al., 2015). This is because short edges can be congested or have lower speed limit. For instance, in the motivating example, we assumed the time of travelling between b2 and b1 is less than that of travelling from b3 to b1, this

 https://doi.org/10.12928/commicast.v5i1.9955

is because R2 is shorter than R4. On the other hand, calculating the exact travel time between the broken down bus and other busses at query time is timely expensive.

Accordingly, we used a cost estimation framework to com pute the travel cost on the road network when processing a MKNNsCC query. The framework is based on precomputing and storing the cost between different geographical zones in the road network and retrieving the stored cost at the query time. Splitting the road network into multiple geographical zones can be performed with the assistance of any spatial data structure (e.g., Quadtrees, Octrees, R-tress). In this paper, we used the quadtree, which is two-dimensional data structure generally used in image processing and spatial indexing (Shahabi et al., 2002).

The straightforward method to index a road network using a quadtree is to index the network vertices in the leaves of the quadtree based on the desired density level. Next, we precompute and store the travel time between every pair of leaf nodes to use it at the query time. However, since most of the road network used in this paper is composed of large road segments (e.g., rural roads connecting villages), it might not be efficient to only rely on the network vertices. This is because the point of indexing the network is to precompute and store the estimated travel time between any two points, which is not achieved when indexing the vertices of rural road network.

Therefore, we solve this problem by adding new network vertices on large road segments. our implementation starts by defining a maximum segment cost variable SEGMax, which defines if a road segment requires an extra vertex. When the segment cost is larger than that of SEGMax we add a new vertex halfway the cost of the segment. After balancing the road network by adding all required vertices, we index the road network vertices into the leave nodes based on the desired density level. Next, we compute and store the time of travelling between every pair of vertices in the road network using any best first search algorithm (Dijkstra, 1959).

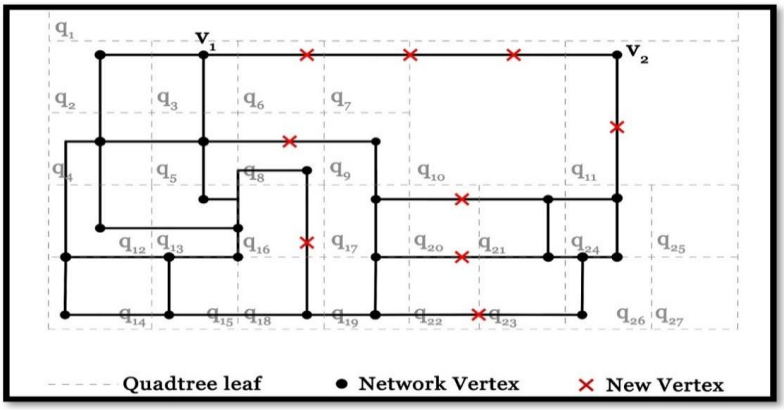


Fig. 3. Road Network Indexing with Quadtree

For example, in Figure 2, first we add the network new vertices whenever a segment cost exceeded the variable SEGMax, such as the segment [V(1),V2], which needed to be divided three

times. Next, we index the network vertices into the quadtree leaves based on the desired vertices density (e.g., 2). Finally, we run Dijkstra's algorithm from every vertex in the network to find and store the average travel time between every pair of leaves. In order to obtain the travel cost between a pair of buses at the MKNNsCC query processing time, we retrieve the average travel time between the quadtree leaf nodes containing the buses.

The processing time and memory cost of precomputing and storing this framework is extremely sensitive to the maximum density of quadtree nodes. Nevertheless, as the framework is precomputed offline, the processing time should not be a concern. Additionally, the memory cost of storing the precomputed average travel time between the quadtree leaves can be tolerated when choosing the suitable density level at the leaves. As will be displayed in Section 5 of this research, the less vertices we store at each quadtree leaf, the more precise travel time we obtain.

Moreover, the less the value of the variable SEGMax, the more density level needed as well as the more accurate results we achieve. Since we are using a dynamic road network with changing traffic conditions throughout the day, we can build this framework based on the buses historical data at different time slots of during the day. a) Baseline Algorithm:: In the baseline algorithm, we used the above cost estimation framework to estimate the cost between any pair of buses. Next, we find every possi ble combination of buses with possible capacity (bic > 1) excluding the query bus. In addition, the maximum number of the combination set must not be greater than the broken bus number of passengers CdB ≤ bxp. Then, we sort the combination based on their total travel cost to the query bus. Finally, we pick the combination with the lowest cost as long as its total passenger capacity is equal or more than that of the query bus.

4. Result and Discussion

Framework Evaluation

In this section we investigate the performance of both travel cost estimation framework and the Bus* algorithm in terms of both effectiveness and efficiency. We performed our experiments on a desktop PC with 32GB RAM and a 3.8GHz Intel® Core™ i7 CPU. The size of the page is 4K bytes. We used the GPS data of a group of 114 buses operated by Shaqra University during a period of over 8 months. The average number of GPS points per bus is 653742.

The University uses these busses to transport some students from their villages towards the university main campuses back and forth. We also used the road network of GCC States extracted from Open Street Map with over 18 million vertices. However, we extracted the minimum bounding rectangle (MBR) of the area covered by the buses and used it for the experiments. Weadjust the experiments parameters such as the maximum length of a network edge, the number of vertices in

 https://doi.org/10.12928/commicast.v5i1.9955

the Quadtree leaf nodes, the buses occupancy and density on the road network to obtain a deep understanding of the framework and algorithm performance.

In each of the following experi ments, we detail the different settings. As demonstrated in Section 3.2, the main goal of using the framework is to avoid the expensive computing of real travel cost on the road network at the query time. Therefore, the framework retrieves the precomputed estimate travel cost between a pair of buses based on the historical data. In order to rely on this framework, we need to evaluate its performance through the following experiments.

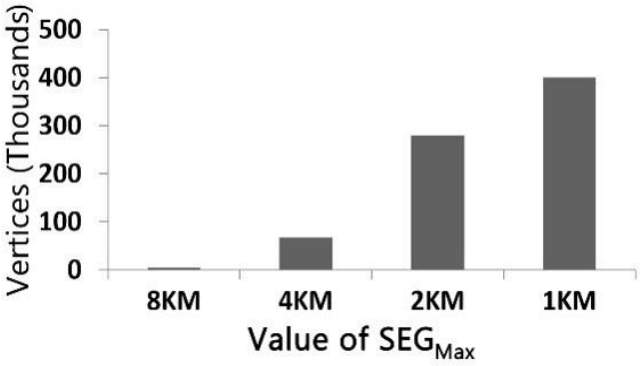


Fig. 4. This figure show that increasing the number of reflecting elements of the IRS improves the secrecy capacity of the system

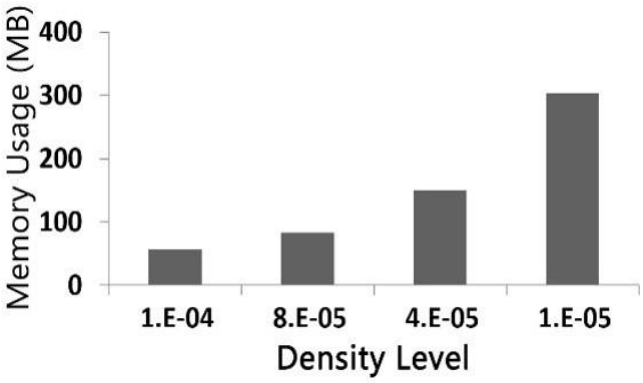


Fig. 5. This figure shows that increasing the number of reflecting elements of the IRS improves the secrecy capacity of the system

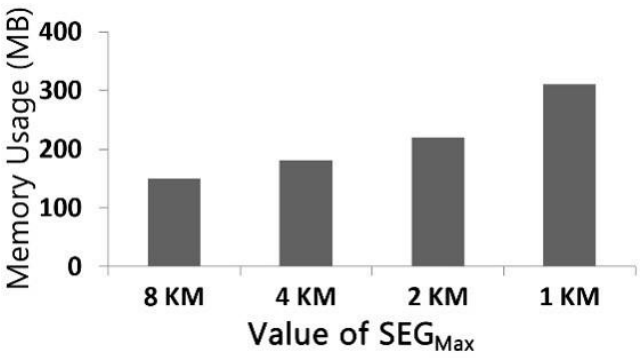


Fig. 6. This figure shows that increasing the number of reflecting elements of the IRS improves the secrecy capacity of the system

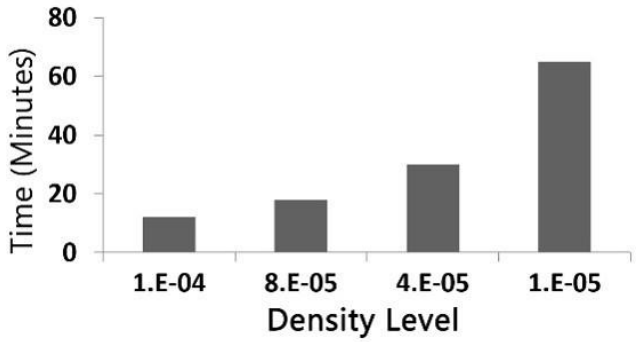


Fig. 7. This figure shows that increasing the number of reflecting elements of the IRS improves the secrect capacity of the system

Framework running time: Fig 8 illustrates the increase of the framework construction time while we vary the Quadtree density level from 0.0001% to 0.00001%. This is because the less dense the leaf nodes, the more nodes required, thus, the more time demanded to construct the framework. At the least density level, the required time is around 1 hour, which is acceptable as the framework is constructed offline.

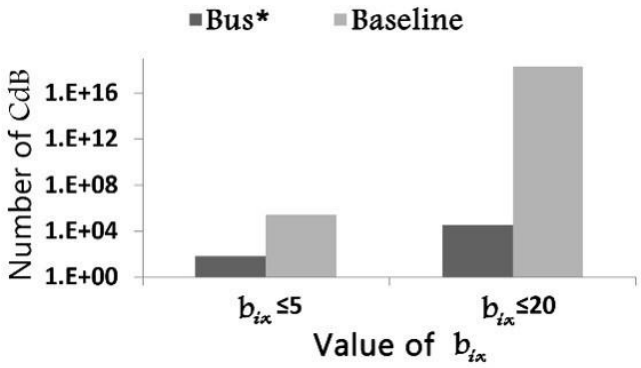


Fig. 8. Bus* Running Stage

Framework Construction: As detailed at the framework implementation section, indexing the vertices of rural road network into the quadtree might not be effective to estimate the travel time between any two points on the road network. Thus, we need the maximum segment cost variable SEGmax, which decides when to add a new vertex. Road network vertices number: the number of vertices in the road network is highly affected by the variable SEGmax. This is because the smaller the value of SEGmax, the more vertices we need to add to the road network. Fig 4 illustrates the increase of the number of road network vertices as the value of the variable SEGmax decreases from 8KM to 1 KM. As can be seen the number of extra needed vertices increases from 50000 vertices when SEGmax=8km to 400000 vertices when SEGmax=1km.

This is well justified knowing most of the vertices are located within major cities, thus the average distance between them is usually less than SEGmax. Framework memory cost: the purpose of utilizing the frame work is to store the pre-computed travel time between the Quadtree leaf nodes. Therefore, the number of needed values to store is n2, where n is the number of Quadrees

https://doi.org/10.12928/commicast.v5i1.9955

nodes. Fig 5 illustrates the required space to store the framework while varying the road network vertices density level at the Quadtree nodes from 0.0001% to 0.00001% of total number of vertices in the network, while we fix the value of SEGmax to 2km.

It can be seen from the figure that, when the density level decreases, when need more nodes in the framework, thus, the memory cost increases. The framework requires around 300 MB to be stored when the density level is 0.00001% which is considered a very low memory consumption. On the other hand, in Fig6 we evaluated the memory consumption while f ixing the density level to 0.00003% and changing the value of SEGmax from 8 KM to 1KM. As shown in the figure, the less the value of SEGmax the more memory the framework consumes. Nevertheless, the variation in the value of SEGmax does not majorly affect the memory consumption as discussed in the prior experiment.

Bus* Evaluation

Since both the baseline and Bus* algorithms are using the same framework and accurately answer the MKNNsCC query, there is no need to compare their accuracy. However, in this section we will compare their performance. The Number of processed candidate sets: As discussed in Section 4, the novelty of and Bus* relied on its ability to terminates after processing small number of possible candidates. Fig 8 illus trates the average number of candidate sets processed for a random 100 query. It can be clearly seen that Bus* only need to process less than 5% of the sets processed by the baseline algorithm when the number of passengers of the broken bus is less than 5 (bix ≤ 5). However, as bix reaches 20, Bus* processes less than 1% of the sets processed by the baseline algorithm.

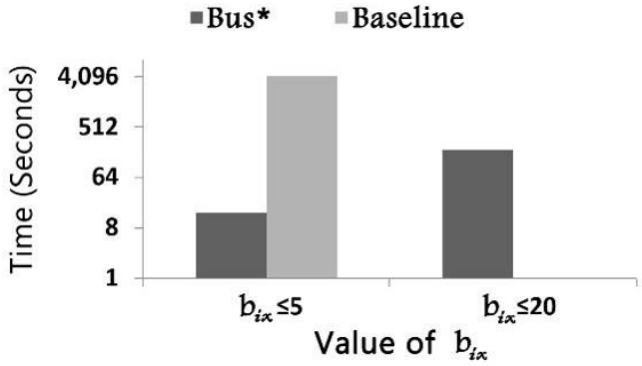


Fig. 9. Bus* Running Stage

Processing Time: As can be seen from Fig 9 the average time needed for Bus* to process the MKNNsCC query is significantly less than that of the baseline algorithm. For example, when (bix ≤ 5), Bus* in average takes a few seconds to process the MKNNsCC query, while the base line algorithm takes around an hour. However, when (bix ≤ 20) Bus* in average takes 30 minutes while the baseline algorithm could not finish even after 24 hours.

5. Conclusion

In this work, we defined a new problem the MKNNsCC query, which finds the k nearest neighbors while considering 9 the capacity constraint. We utilized a road network cost estimation framework based on Quadtree indexing. We also proposed a novel A∗ inherited algorithm named Bus*, which solves the MKNNsCC query efficiently. The Bus* algorithm is run over a virtual fully connected graph connecting all candidate objects in the network with extraordinary edges. The main advantage of this algorithm is its ability to terminate the search for the optimal set of buses when there is no further promising set. As shown in the experiments section, the Bus* algorithm showed efficient performance as well as effective results when evaluated over real dataset. Furthermore, by designing the path for the low-power Adhoc network, its effectiveness can be further examined in future work. This work can be modified in several ways. For example, some enabling technologies such intelligent reflecting surfaces can be integrated to further enhance the system performance. Moreover, learning techniques and algorithm can also be adopted in our future studies.

References

Ahmed, M., Khan, W. U., Ihsan, A., Li, X., Li, J., & Tsiftsis, T. A. (2022). Backscatter Sensors Communication for 6G Low-Powered NOMA-Enabled IoT Networks Under Imperfect SIC. *IEEE Systems Journal*, *16*(4), 5883–5893. https://doi.org/10.1109/JSYST.2022.3194705

Ahmed, M., Raza, S., Mirza, M. A., Aziz, A., Khan, M. A., Khan, W. U., Li, J., & Han, Z. (2022). A survey on vehicular task offloading: Classification, issues, and challenges. *Journal of King Saud University - Computer and Information Sciences*, *34*(7), 4135–4162. https://doi.org/10.1016/j.jksuci.2022.05.016

Ali, Z., Farooq, W., Khan, W. U., Qureshi, M., & Sidhu, G. A. S. (2021). Artificial intelligence techniques for rate maximization in interference channels. *Physical Communication*, *47*. https://doi.org/10.1016/j.phycom.2021.101294

Ali, Z., Khan, W. U., Ihsan, A., Waqar, O., Sidhu, G. A. S., & Kumar, N. (2021). Optimizing Resource Allocation for 6G NOMA-Enabled Cooperative Vehicular Networks. *IEEE Open Journal of Intelligent Transportation Systems*, *2*, 269–281. https://doi.org/10.1109/OJITS.2021.3107347

Ali, Z., Khan, W. U., Sardar Sidhu, G. A., K, N., Li, X., Kwak, K. S., & Bilal, M. (2022). Fair power allocation in cooperative cognitive systems under NOMA transmission for future IoT networks. *Alexandria Engineering Journal*, *61*(1), 575–583. https://doi.org/10.1016/j.aej.2021.04.107

Aljubayrin, S., He, Z., & Zhang, R. (2015). Skyline trips of multiple POIs categories. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9050*, 189–206. https://doi.org/10.1007/978-3-319-18123-3_12

Aljubayrin, S., Qi, J., Jensen, C. S., Zhang, R., He, Z., & Wen, Z. (2015). The safest path via safe zones. *Proceedings - International Conference on Data Engineering*, *2015-May*, 531–542. https://doi.org/10.1109/ICDE.2015.7113312

Asif, M., Ihsan, A., Khan, W. U., Ranjha, A., Zhang, S., & Wu, S. X. (2023). Energy-Efficient Beamforming and Resource Optimization for AmBSC-Assisted Cooperative NOMA IoT Networks. *IEEE Internet of Things Journal*, *10*(14), 12434–12448. https://doi.org/10.1109/JIOT.2023.3247021

Assegaff, S. B., & Pranoto, S. O. (2020). Price Determines Customer Loyalty in Ride-Hailing Services. *American Journal of Humanities and Social Sciences Research*, *3*.

Athitsos, V., Alon, J., & Sclaroff, S. (2005). Efficient nearest neighbor classification using a cascade

 https://doi.org/10.12928/commicast.v5i1.9955

of approximate similarity measures. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, *I*, 486–493. https://doi.org/10.1109/CVPR.2005.141

Basu, S., Karki, M., Ganguly, S., DiBiano, R., Mukhopadhyay, S., & Nemani, R. (2015). Learning sparse feature representations using probabilistic quadtrees and Deep Belief Nets. *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2015 - Proceedings*, 367–372. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84961832113&partnerID=40&md5=7d6cf0813cb5c9ea539809d7428b1907

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

Duch, A., & Martinez, C. (2005). Improving the Performance of Multidimensional Search Using Fingers. *ACM Journal of Experimental Algorithmics*, *10*, 2.4. https://doi.org/10.1145/1064546.1180615

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Hasan, T., Malik, J., Bibi, I., Khan, W. U., Al-Wesabi, F. N., Dev, K., & Huang, G. (2023). Securing Industrial Internet of Things Against Botnet Attacks Using Hybrid Deep Learning Approach. *IEEE Transactions on Network Science and Engineering*, *10*(5), 2952–2963. https://doi.org/10.1109/TNSE.2022.3168533

Hautamäki, V., Kärkkäinen, I., & Fränti, P. (2004). Outlier detection using k-nearest neighbour graph. *Proceedings - International Conference on Pattern Recognition*, *3*, 430–433. https://doi.org/10.1109/ICPR.2004.1334558

Huang, X., Jensen, C. S., Lu, H., & Šaltenis, S. (2007). S-GRID: A versatile approach to efficient query processing in spatial networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4605 LNCS*, 93–111. https://doi.org/10.1007/978-3-540-73540-3_6

Ihsan, A., Chen, W., Asif, M., Khan, W. U., Wu, Q., & Li, J. (2022). Energy-Efficient IRS-Aided NOMA Beamforming for 6G Wireless Communications. *IEEE Transactions on Green Communications and Networking*, *6*(4), 1945–1956. https://doi.org/10.1109/TGCN.2022.3209617

Ihsan, A., Chen, W., Khan, W. U., Wu, Q., & Wang, K. (2023). Energy-Efficient Backscatter Aided Uplink NOMA Roadside Sensor Communications Under Channel Estimation Errors. *IEEE Transactions on Intelligent Transportation Systems*, *24*(5), 4962–4974. https://doi.org/10.1109/TITS.2023.3240159

Jameel, F., Khan, W. U., Shah, S. T., & Ristaniemi, T. (2019). Towards intelligent IoT networks: Reinforcement learning for reliable backscatter communications. *2019 IEEE Globecom Workshops, GC Wkshps 2019 - Proceedings*. https://doi.org/10.1109/GCWkshps45667.2019.9024401

Jan, M., Soomro, S. A., & Ahmad, N. (2017). Impact of Social Media on Self-Esteem. *European Scientific Journal, ESJ*, *13*(23). https://doi.org/10.19044/esj.2017.v13n23p329

Jensen, C. S., Kolář, J., Pedersen, T. B., & Timko, I. (2003). Nearest neighbor queries in road networks. *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, 1–8. https://doi.org/10.1145/956676.956677

Khan, A. U., Abbas, G., Abbas, Z. H., Bilal, M., Shah, S. C., & Song, H. (2022). Reliability Analysis of Cognitive Radio Networks with Reserved Spectrum for 6G-IoT. *IEEE Transactions on Network and Service Management*, *19*(3), 2726–2737. https://doi.org/10.1109/TNSM.2022.3168669

Khan, A. U., Tanveer, M., Khan, W. U., Nebhen, J., Li, X., Zeng, M., & Dobre, O. A. (2021). An enhanced spectrum reservation framework for heterogeneous users in CR-Enabled IoT Networks. *IEEE Wireless Communications Letters*, *10*(11), 2504–2508. https://doi.org/10.1109/LWC.2021.3105728

Khan, W. U. (2019). Maximizing physical layer security in relay-assisted multicarrier nonorthogonal multiple access transmission. *Internet Technology Letters*, *2*(2). https://doi.org/10.1002/itl2.76

Khan, W. U., Ali, Z., Lagunas, E., Mahmood, A., Asif, M., Ihsan, A., Chatzinotas, S., Ottersten, B., & Dobre, O. A. (2023). Rate Splitting Multiple Access for Next Generation Cognitive Radio Enabled LEO Satellite Networks. *IEEE Transactions on Wireless Communications*, *22*(11), 8423–8435. https://doi.org/10.1109/TWC.2023.3263116

Khan, W. U., Ali, Z., Waqas, M., & Sidhu, G. A. S. (2019). Efficient power allocation with individual QoS guarantees in future small-cell networks. *AEU - International Journal of Electronics and Communications*, *105*, 36–41. https://doi.org/10.1016/j.aeue.2019.03.016

Khan, W. U., Ihsan, A., Nguyen, T. N., Ali, Z., & Javed, M. A. (2022). NOMA-Enabled Backscatter Communications for Green Transportation in Automotive-Industry 5.0. *IEEE Transactions on Industrial Informatics*, *18*(11), 7862–7874. https://doi.org/10.1109/TII.2022.3161029

Khan, W. U., Imtiaz, N., & Ullah, I. (2021). Joint optimization of NOMA-enabled backscatter communications for beyond 5G IoT networks. *Internet Technology Letters*, *4*(2). https://doi.org/10.1002/itl2.265

Khan, W. U., Jameel, F., Li, X., Bilal, M., & Tsiftsis, T. A. (2021). Joint Spectrum and Energy Optimization of NOMA-Enabled Small-Cell Networks with QoS Guarantee. *IEEE Transactions on Vehicular Technology*, *70*(8), 8337–8342. https://doi.org/10.1109/TVT.2021.3095955

Khan, W. U., Jameel, F., Sidhu, G. A. S., Ahmed, M., Li, X., & Jantti, R. (2020). Multiobjective Optimization of Uplink NOMA-Enabled Vehicle-to-Infrastructure Communication. *IEEE Access*, *8*, 84467–84478. https://doi.org/10.1109/ACCESS.2020.2991197

Khan, W. U., Jamshed, M. A., Lagunas, E., Chatzinotas, S., Li, X., & Ottersten, B. (2023). Energy Efficiency Optimization for Backscatter Enhanced NOMA Cooperative V2X Communications Under Imperfect CSI. *IEEE Transactions on Intelligent Transportation Systems*, *24*(11), 12961–12972. https://doi.org/10.1109/TITS.2022.3187567

Khan, W. U., Jamshed, M. A., Mahmood, A., Lagunas, E., Chatzinotas, S., & Ottersten, B. (2022). Backscatter-Aided NOMA V2X Communication under Channel Estimation Errors. *IEEE Vehicular Technology Conference*, *2022-June*. https://doi.org/10.1109/VTC2022-Spring54318.2022.9860382

Khan, W. U., Lagunas, E., Ali, Z., Javed, M. A., Ahmed, M., Chatzinotas, S., Ottersten, B., & Popovski, P. (2022). Opportunities for Physical Layer Security in UAV Communication Enhanced with Intelligent Reflective Surfaces. *IEEE Wireless Communications*, *29*(6), 22–28. https://doi.org/10.1109/MWC.001.2200125

Khan, W. U., Lagunas, E., Mahmood, A., Ali, Z., Chatzinotas, S., Ottersten, B., & Dobre, O. A. (2022). Integration of Backscatter Communication with Multi-cell NOMA: A Spectral Efficiency Optimization under Imperfect SIC. *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, *2022-Novem*, 147–152. https://doi.org/10.1109/CAMAD55695.2022.9966913

Khan, W. U., Lagunas, E., Mahmood, A., Elhalawany, B. M., Chatzinotas, S., & Ottersten, B. (2022). When RIS Meets GEO Satellite Communications: A New Sustainable Optimization Framework in 6G. *IEEE Vehicular Technology Conference*, *2022-June*. https://doi.org/10.1109/VTC2022-Spring54318.2022.9860805

Khan, W. U., Li, X., Ihsan, A., Ali, Z., Elhalawany, B. M., & Sidhu, G. A. S. (2021). Energy efficiency maximization for beyond 5G NOMA-enabled heterogeneous networks. *Peer-to-Peer Networking and Applications*, *14*(5), 3250–3264. https://doi.org/10.1007/s12083-021-01176-5

Khan, W. U., Liu, J., Jameel, F., Khan, M. T. R., Ahmed, S. H., & Jantti, R. (2020). Secure backscatter communications in multi-cell NOMA Networks: Enabling link security for massive IoT

networks. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2020*, 213–218. https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162938

Khowaja, S. A., Khuwaja, P., Dev, K., Lee, I. H., Khan, W. U., Wang, W., Qureshi, N. M. F., & Magarini, M. (2023). A Secure Data Sharing Scheme in Community Segmented Vehicular Social Networks for 6G. *IEEE Transactions on Industrial Informatics*, *19*(1), 890–899. https://doi.org/10.1109/TII.2022.3188963

Li, J., Fang, H. Y., Ma, Y. R., & Yang, H. B. (2014). Research on point cloud data management based on spatial index and database. *Advanced Materials Research*, *850–851*, 685–688. https://doi.org/10.4028/www.scientific.net/AMR.850-851.685

Lopac, V., Brant, S., & Paar, V. (1986). Level density fluctuations and characterization of chaos in the realistic model spectra for odd-odd nuclei. *Zeitschrift Für Physik A Hadrons and Nuclei*, *356*(2), 113–118. https://doi.org/10.1007/s002180050156

Mahmood, A., Ahmed, A., Naeem, M., Amirzada, M. R., & Al-Dweik, A. (2022). Weighted utility aware computational overhead minimization of wireless power mobile edge cloud. *Computer Communications*, *190*, 178–189. https://doi.org/10.1016/j.comcom.2022.04.017

Mahmood, A., Ahmed, A., Naeem, M., & Hong, Y. (2020). Partial offloading in energy harvested mobile edge computing: A direct search approach. *IEEE Access*, *8*, 36757–36763. https://doi.org/10.1109/ACCESS.2020.2974809

Mahmood, A., Hong, Y., Ehsan, M. K., & Mumtaz, S. (2021). Optimal Resource Allocation and Task Segmentation in IoT Enabled Mobile Edge Cloud. *IEEE Transactions on Vehicular Technology*, *70*(12), 13294–13303. https://doi.org/10.1109/TVT.2021.3121146

Mahmood, A., Vu, T. X., Khan, W. U., Chatzinotas, S., & Ottersten, B. (2022). Optimizing Computational and Communication Resources for MEC Network Empowered UAV-RIS Communication. *2022 IEEE GLOBECOM Workshops, GC Wkshps 2022 - Proceedings*, 974–979. https://doi.org/10.1109/GCWkshps56602.2022.10008627

Matke, M., Saurabh, K., & Singh, U. (2023). An Empirical Evaluation of Machine Learning Algorithms for Intrusion Detection in IIoT Networks. *2023 IEEE 20th India Council International Conference, INDICON 2023*, 1353–1358. https://doi.org/10.1109/INDICON59947.2023.10440779

Nutanong, S., Zhang, R., Tanin, E., & Kulik, L. (2009). V*-kNN: An efficient algorithm for moving k nearest neighbor queries. *Proceedings - International Conference on Data Engineering*, 1519–1522. https://doi.org/10.1109/ICDE.2009.63

Petrescu-Mag, R. M., Vermeir, I., Petrescu, D. C., Crista, F. L., & Banatean-Dunea, I. (2020). Traditional foods at the click of a button: The preference for the online purchase of romanian traditional foods during the COVID-19 pandemic. *Sustainability (Switzerland)*, *12*(23). https://doi.org/10.3390/su12239956

Rasheed, I., Asif, M., Ihsan, A., Khan, W. U., Ahmed, M., & Rabie, K. M. (2023). LSTM-Based Distributed Conditional Generative Adversarial Network for Data-Driven 5G-Enabled Maritime UAV Communications. *IEEE Transactions on Intelligent Transportation Systems*, *24*(2), 2431–2446. https://doi.org/10.1109/TITS.2022.3187941

Raza, S., Wang, S., Ahmed, M., Anwar, M. R., Mirza, M. A., & Khan, W. U. (2022). Task Offloading and Resource Allocation for IoV Using 5G NR-V2X Communication. *IEEE Internet of Things Journal*, *9*(13), 10397–10410. https://doi.org/10.1109/JIOT.2021.3121796

Shahabi, C., Kolahdouzan, M. R., & Sharifzadeh, M. (2002). A road network embedding technique for K-Nearest Neighbor search in moving object databases. *Proceedings of the ACM Workshop on Advances in Geographic Information Systems*, 94–100. https://doi.org/10.1145/585147.585167

Shen, B., Zhao, Y., Li, G., Zheng, W., Qin, Y., Yuan, B., & Rao, Y. (2017). V-Tree: Efficient KNN search on moving objects with road-network constraints. *Proceedings - International Conference on*

Data Engineering, 609–620. https://doi.org/10.1109/ICDE.2017.115

Sparrow, B. H. (2004). Projections of Power: Framing News, Public Opinion, and U.S. Foreign Policy. *Perspectives on Politics*. https://doi.org/10.1017/s1537592704350589

Tianyang, D., Lulu, Y., Qiang, C., Bin, C., & Jing, F. (2019). Direction-aware KNN queries for moving objects in a road network. *World Wide Web*, *22*(4), 1765–1797. https://doi.org/10.1007/s11280-019-00657-1

Wang, S., Gao, S., Feng, X., Murray, A. T., & Zeng, Y. (2018). A context-based geoprocessing framework for optimizing meetup location of multiple moving objects along road networks. *International Journal of Geographical Information Science*, *32*(7), 1368–1390. https://doi.org/10.1080/13658816.2018.1431838

Yu, S., Khan, W. U., Zhang, X., & Liu, J. (2021). Optimal power allocation for NOMA-enabled D2D communication with imperfect SIC decoding. *Physical Communication*, *46*. https://doi.org/10.1016/j.phycom.2021.101296

Zhao, X., Yu, J., Ge, X., & Hao, R. (2024). Towards efficient Secure Boolean Range Query over encrypted spatial data. *Computers and Security*, *136*. https://doi.org/10.1016/j.cose.2023.103544

Zheng, Z., & Su, D. (2014). Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. *Transportation Research Part C: Emerging Technologies*, *43*, 143–157. https://doi.org/10.1016/j.trc.2014.02.009

 https://doi.org/10.12928/commicast.v5i1.9955